(appeared in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 11, pp. 1186-1205, November 1998)

Shape Evolution with Structural and Topological Changes using Blending

Douglas DeCarlo* and Dimitris Metaxas†

Abstract

This paper describes a framework for the estimation of shape from sparse or incomplete range data. It uses a shape representation called blending, which allows for the geometric combination of shapes into a unified model—selected regions of the component shapes are cut-out and glued together. Estimation of shape using this representation is realized using a physics-based framework, and also includes a process for deciding how to adapt the structure and topology of the model to improve the fit. The blending representation helps avoid abrupt changes in model geometry during fitting by allowing the smooth evolution of the shape, which improves the robustness of the technique. We demonstrate this framework with a series of experiments showing its ability to automatically extract structured representations from range data given both structurally and topologically complex objects.

1 Introduction

Work on shape estimation exhibits a trade-off between conciseness and expressiveness in representation. The choice of shape representations must be made carefully—especially given data that represents an object of complex topology, or given data that is sparse or incomplete. In this paper, a shape representation called *blending* is described, which allows for the combination of two shapes into a single model. The component shapes are "cut" apart, and the selected pieces are "glued" together. The gluing is realized geometrically using a method of interpolation. A shape estimation framework which uses blended shapes is used to *automatically* fit a surface to a given set of range data. This range data can be sparse or incomplete, and can represent an object of arbitrary topology. The fitting is made more robust by avoiding any large geometric jumps using smooth *evolution* of the model. The basic model fitting is carried out using a physics-based estimation framework [1, 2]. This framework is augmented with a geometric decision process that allows parts and holes to be added to the model.

1.1 Situating blending

Blending has three advantages, which we outline here. First, blending offers a concise and flexible representation of shape, which facilitates its use in recognition.

Considering the spectrum of shape estimation work, at one end are models with a small number of parameters such as generalized cylinders [3, 4, 5], geons [6], superquadrics [7, 8], hyperquadrics [9, 10], and algebraic surfaces [11]. The small parameter sets make these models useful for recognition applications, but they have a fairly limited shape coverage, and can only represent objects of fixed topology.

At the other extreme are representations with many parameters, such as free-form surfaces [12, 13, 14, 15, 16], advancing front techniques [17, 18] and particles [19]. These methods obtain a wide shape coverage at the expense of the abstraction required for recognition tasks. Some of these methods allow for the modeling of surfaces of complex topology [12, 13, 17, 15, 19, 18], but require complete and dense data of an object in order to achieve an acceptable level of robustness. These methods model topological changes using local surface information only.

In the middle are models which aim to strike a balance between extremes—blending is one of them. We introduced a simple form of blending in [20, 21] and its generalization in [22]. Blending enables the combination of multiple globally defined shapes into a single model. Selected parts of the models are connected together using a method of interpolation. These parts are added on either to increase the shape coverage of the model, or to add holes through the object to form complex topologies. Other middle-level models have used a combination of global and local deformations [1, 23], or a set of deformations with parameters ordered by level of detail [24, 25]. However, while the global parameters in [1, 23] or the coarse scale parame-

*D. DeCarlo is with the Department of Computer Science and Center for Cognitive Science, Rutgers University, 110 Frelinghuysen Road, Piscataway, NJ 08854-8019. E-mail: decarlo@cs.rutgers.edu.

†D. Metaxas is with the Department of Computer and Information Science, University of Pennsylvania, 200 S. 33rd St, Philadephia, PA 19104-6389. E-mail: dnm@central.cis.upenn.edu.

ters in [24, 25] capture the rough shape of an object, these methods do not deal with an object consisting of a number of distinct parts so elegantly. In addition, these methods only deal with surfaces of simple topology.

Enabling smooth shape evolution is the second advantage of blending. The evolution of a shape model describes how it changes over time. From the point of view of extraction, the passage of time constitutes the sequence of model shapes generated by the estimation process. In this case, the time is discretized into a sequence of events. Iterative fitting processes usually make small improvements to the model from one time-step to the next. This proceeds smoothly until a local minimum is reached. At this point, large changes to the model are applied, which can be representational, geometric, or topological. For example, a representation capable of producing a similar class of shapes to blending was introduced in [14]; this representation instead uses Bézier surfaces to connect the parts together. This system permits topological changes during estimation; however, the model changes are abrupt—in one step, a boolean (CSG) shape operation will add on a part, or cut a hole though the shape. While this abrupt action often leads to the desired global minimum, it can also be the source of a lack of robustness.

In most cases, the metric used to judge the quality of a particular solution is geometric in nature. These metrics also tend to be only locally effective for comparison—the most meaningful comparison between two models is for those that differ geometrically by only a small amount. Therefore, a good goal is to minimize (or eliminate) the abrupt geometric alteration during large changes in the model. Blending achieves this goal by producing geometrically smooth transformations for both structural changes to the model (such as the addition of a part), and the alteration of the model topology. In [22], an additional part or hole appears gradually in shape evolution. Changes in representation or topology are not a problem. This is a key motivation in the design of the blending representation for use in shape estimation.

Third, blending facilitates the extraction of a *structured* representation of shape. A structured representation is constructed by breaking down a shape into primitive components that have meaning in terms of the geometric structure of the object. Blending can specify the composition of component shapes in this way. There is thus an overlap between shape estimation techniques which extract a structured representation, and methods for part segmentation. The main distinction is that in shape estimation there is no strict definition of what constitutes a "part", as is often the case in part segmentation work [26]. A further distinction is that most shape estimation techniques combine attached parts using a single surface instead of leaving them as a collection of (perhaps overlapping) pieces.

There is, of course, no single correct answer to part segmentation [27]; approaches for the part segmentation of range data make the best guess from geometric information alone. As [28] has observed, part segmentation algorithms can be classified as to whether models are merged together (local-to-global) in a geometrically bottom-up fashion, or are split apart (global-to-local) in order to form the final segmentation.

Much of the existing part segmentation work uses a local-to-global approach where numerous small models are *merged* together into a few larger models. These methods can use surface patches [29] or shape primitives [30, 31, 32, 33]. Other local-to-global approaches use geometric analysis methods in an attempt to find the boundaries between the parts [34, 35, 36]. Some of these analysis methods are formalized in terms of differential geometry [26]. The local-to-global methods do not perform well given incomplete or sparse data, with the exception of [30, 31], which use a minimum description length (MDL) criteria for merging.

Blending is one of the few approaches that fit and segment by splitting in a global-to-local fashion, where a model is repeatedly *split* until a desired accuracy in the fit is achieved [22, 37, 14, 8]. A combination of both approaches is used in [28], where a local-to-global surface segmentation process is used to guide a global-to-local volumetric primitive extraction process.

1.2 Outline

The work presented here is a more thorough and detailed treatment of the work on shape blending introduced in [22]. Additional experiments have also been performed that demonstrate the robustness and stability of the results.

After some preliminaries in Section 2, blended shapes are introduced in Section 3 followed by a description of smooth shape evolution in Section 4, and its relationship to blending. Section 5 explains how blending and evolution are applied to shape reconstruction, some examples of which are presented in Section 6.

2 Preliminaries

When concerned with topology of surfaces, there is some terminology that needs to be defined. To simplify this discussion, all surfaces described here are three-dimensional surfaces, such as the sphere in Figure 1(b). In particular, they are two-dimensional manifolds–this means that any

small patch of the surface looks (topologically) like a small patch of $\mathbb{R}^2$.

The *topology* of a shape is specified by the connectivity of its surface on a global level. For example, a sphere and torus have different surface topologies. There are a number of ways of quantifying the topology of a surface. One of the most common is the *genus* of a surface, which specifies the number of holes (or handles) that a surface contains–a sphere is genus 0, while a torus is genus 1. Another method of topological classification is whether a surface is *orientable*, which is the case only if the surface has a notion of an "inside" and "outside". A single-sided surface, such as a Klein bottle, is non-orientable.

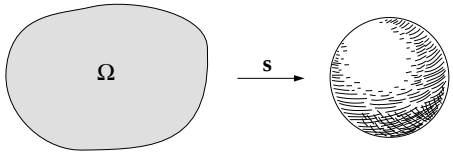


Figure 1: Example shape mapping $\mathbf{s}\colon \Omega \mapsto \mathbb{R}^3$

A topological space Ω, as in Figure 1(a), is endowed with only connectivity information–no geometry. In this paper, topological spaces such as Ω are drawn abstractly for illustrative purposes (and are not necessarily flat disks). The geometry of the shape is specified by associating a point in $\mathbb{R}^3$ with each point in Ω. This association is often performed by a mapping, such as with $\mathbf{s}$ in Figure 1, in which case Ω is the *domain* of the shape. If $\mathbf{s}$ is continuous and invertible, it is called a *homeomorphism* and preserves topology under mapping. So if the mapping $\mathbf{s}$ in Figure 1 is a homeomorphism, then Ω must be topologically equivalent to a sphere.

The surface parameterization of a shape can have a coordinate system imposed over its domain for the purpose of identifying points and directions on the surface. Sometimes, these parameterizations can have singularities. For example, a latitude-longitude parameterization of a sphere has singularities at each pole—points where longitude doesn't matter. A latitude-longitude parameterization of a torus is singularity-free. A surface can have a global parameterization, meaning that there is a single coordinate system for the entire domain, or it can have a local parameterization, meaning that the domain is broken down into a set of (possibly disjoint) regions which cover the entire domain, each of which has a parameterization defined.

3 Blended shapes

In the following section, blending will be described using the physical analogies of "cutting" surfaces apart and "gluing" them together. This surface "surgery" can be used to construct shapes of a particular topology as well as widen the representational coverage of a shape model. In later sections, it will be explained how blending is used as an integral part of the smooth evolution of a model, and how a surface reconstruction process can perform this surface surgery automatically in order to extract a surface from data.

In reading this section, it will help to remember that the key feature of the representation is to permit smooth shape evolution. There will be points where the fitting algorithm decides on changes in representation (see Section 5.4). The model must have parameters that minimize the immediate geometric effects of these changes in representation. The blending representation therefore includes a parameter that seems redundant considering only shape coverage, but is important for describing the continuous change between shapes (this is the parameter *h* described in Section 3.4).

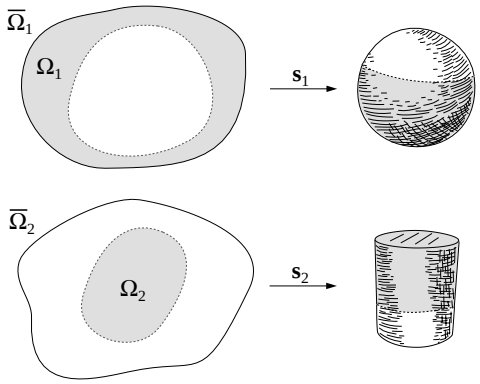


Figure 2: Underlying shapes $\mathbf{s}_1 : \overline{\Omega}_1 \mapsto \mathbb{R}^3$; $\mathbf{s}_2 : \overline{\Omega}_2 \mapsto \mathbb{R}^3$

Blending starts with two *underlying shapes* $\mathbf{s}_1$ and $\mathbf{s}_2$, shown in Figure 2. The shapes $\mathbf{s}_1$ and $\mathbf{s}_2$ are mappings into $\mathbb{R}^3$ that are defined over domains $\overline{\Omega}_1$ and $\overline{\Omega}_2$, respectively. Examples of such shapes include spheres, superquadric ellipsoids, or B-spline surfaces.

Combining $\mathbf{s}_1$ and $\mathbf{s}_2$ to form a blended shape involves the specification of the retained parts of these shapes (cutting), as well as how overlapping parts are connected together (gluing). The retained parts of these shapes are defined by subsets of the shape domains $\Omega_1 \subset \overline{\Omega}_1$ and $\Omega_2 \subset \overline{\Omega}_2$, and are shown in gray on the left of Figure 2. The gray portions on the right of Figure 2 are called the *blending components* of $\mathbf{s}_1$ and $\mathbf{s}_2$. One restriction on the blending components $\mathbf{s}_1$ and $\mathbf{s}_2$ is that they must be homeomorphisms when the domains of $\mathbf{s}_1$ and $\mathbf{s}_2$ are restricted to

Ω_1 and Ω_2 respectively. This will be useful in Section 3.2 when surface parameterizations are constructed.

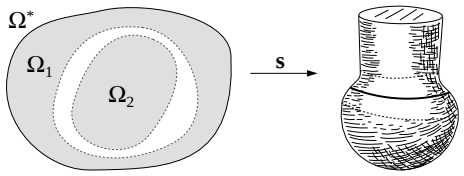


Figure 3: Blended shape $\mathbf{s}\colon \Omega^* \mapsto \mathbb{R}^3$

Figure 3 shows the blended shape $\mathbf{s}$ which is formed using the blending components of $\mathbf{s}_1$ and $\mathbf{s}_2$ showing how certain parts of $\mathbf{s}$ resemble each of its components. The continuous join between the components is formed using interpolation, details of which are provided in Section 3.4. The blended shape $\mathbf{s}$ is defined over a domain Ω^*, which is the result of topologically merging Ω_1 and Ω_2.

The addition of a hole works by removing two disks from the shape, and gluing a hole (a tube) in its place, as in Figure 4. Holes can be added between any two locations on a shape, with the only restriction being the two discarded regions (such as those in Ω_1 in Figure 4) must be disjoint. Adding a hole to a model in a non-abrupt way requires more care, and will be discussed in Section 4.2.

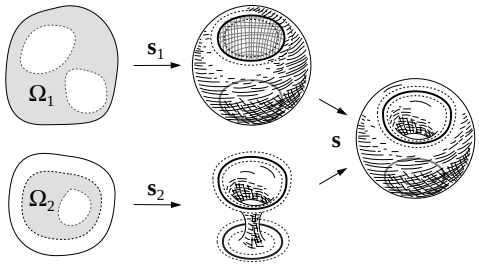


Figure 4: Adding a hole

The next few sections give the details on the cutting and gluing necessary to perform blending.

3.1 Surface surgery

Continuing with the example given above, Figure 5 provides a more detailed look at the cutting and gluing. The curves along which the shapes are cut, $\kappa_1 \subset \Omega_1$ and $\kappa_2 \subset \Omega_2$, are shown as dark lines in Figure 5. The faded regions of the shapes in Figure 5 indicate the portion that is discarded by the cutting.

The neighborhoods around these curves $\omega_1 \subset \Omega_1$ and $\omega_2 \subset \Omega_2$, which are the light gray domain regions in Figure 5, are used to glue the pieces together. Once glued, these strips will overlap both topologically (in the domain) and geometrically. The formulation of the domain overlap is described in Section 3.2, and the geometric overlap (interpolation) is described in Section 3.4.

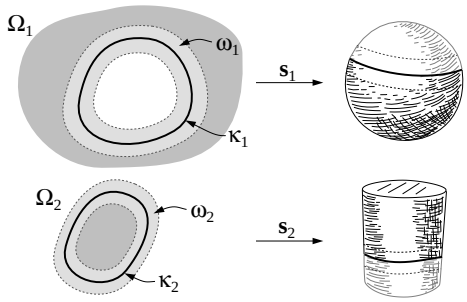


Figure 5: Component shapes $\mathbf{s}_1\colon \Omega_1 \mapsto \mathbb{R}^3$; $\mathbf{s}_2\colon \Omega_2 \mapsto \mathbb{R}^3$

The process of gluing requires a correspondence between ω_1 and ω_2, which is specified using the homeomorphism $\beta\colon \omega_1 \mapsto \omega_2$. Figure 6 illustrates the correspondence mapping β, which is used to place κ_1 and κ_2 into correspondence, and match up the discarded boundary of ω_1 with the retained boundary of ω_2 (and vice versa). Implementation details for the correspondence mapping β are given in Appendix A.1.

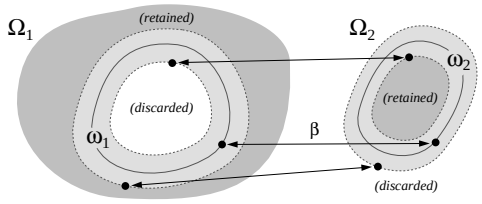


Figure 6: Annulus-shaped regions with correspondence β

For simple cutting and gluing, ω_1 and ω_2 are annulus-shaped (ring-shaped) regions, such as in Figures 5 and 6. In some cases, a correspondence is also established between *all* of Ω_2, and the discarded portion of $\overline{\Omega}_1$, as in Figure 7 (so that $\Omega_2 = \omega_2$). This will allow the contribution of the discarded portion of Ω_1 to the overall shape to be smoothly blended away (more on this in Section 4).

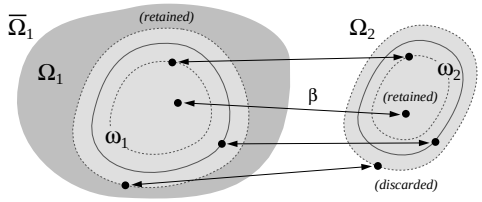


Figure 7: Disk-shaped regions with correspondence β

3.2 Domain gluing

The mapping β is used to glue the component domains together over ω_1 and ω_2. Using the union of the domains $\Omega_1 \bigcup \Omega_2$ as the domain of $\mathbf{s}$ produces doubly covered points by ω_1 and ω_2.

The component domains are properly glued together by applying a notion from topology known as a quotient space [38]. A quotient space is produced when points in a topological space are identified (glued) using an equivalence relation. As an example, consider a topological space which is a line segment; an equivalence relation where only the two endpoints of the segment are equal produces a quotient space which is topologically a closed loop.

An appropriate equivalence relation that glues the domains ω_1 and ω_2 together is the one induced by β:

$$\mathbf{u}_1 \sim \mathbf{u}_2 \iff \beta(\mathbf{u}_1) = \mathbf{u}_2 \text{ and } \mathbf{u}_1 \in \omega_1, \mathbf{u}_2 \in \omega_2 \qquad (1)$$

The domain for the blended shape is written as:

$$\Omega^* = \left(\Omega_1 \bigcup \Omega_2\right) / \sim \qquad (2)$$

where the merged component domains are divided ($/$) by the relation $\sim$, defined by (1).

This method of gluing can be used for the construction of a parameterization over the entire blended shape. It is not a global parameterization, however, but instead consists of two overlapping parameterizations. The region of overlap between the component domains, along with some bookkeeping operations, is as flexible as a global parameterization for the entire surface (at least for the applications here). In [22], a version of blending is presented which instead establishes a global parameterization (which perhaps includes some singularities). In a similar spirit, Grimm and Hughes [39] have described a method for modeling surfaces by gluing together overlapping B-spline patches.

This gluing avoids problems associated with global surface parameterizations. A two-dimensional coordinate system which is placed over a surface of arbitrary genus contains a number of singularities, the number of which is related to the genus of the surface. This is known colloquially as the "hairy ball" theorem [40], which states that a combed hairy ball must have a bald spot, and is a consequence of the Poincaré-Hopf theorem [38].

These singularities would make the specification of surface curves such as κ particularly difficult (from an implementation point of view). Only a disk, open tube, or torus can be parameterized without singularity. Hence, each blending component must be constructed from one of these for it to have a singularity-free two-dimensional parameterization.

3.3 Mesh representation

The surfaces described here are realized using a polygon mesh. The domains of each blended component shape are triangulated separately. The cutting and gluing operations described here present the problem of forming a unified mesh for the entire blended shape. For the shape estimation applications here, however, a single component polygon mesh for the entire shape is not necessary.

The component meshes are constructed by simply omitting the polygonal faces of the original mesh that are completely contained within the discarded portion of the domain. This produces a shape made up of spatially overlapping polygon meshes. Should a single mesh be required, mesh clipping and merging methods can be employed [41].

3.4 Surface interpolation

At this point, the geometry of the gluing process is all that is needed to define $\mathbf{s}\colon \Omega^* \mapsto \mathbb{R}^3$. When forming a blended shape, the retained portions of the surface outside of the blending region, $\Omega_1 \backslash \omega_1$ and $\Omega_2 \backslash \omega_2$, are not altered. In Figure 3, these regions correspond to the shape areas below and above the dotted lines. The connection between these two pieces is created using linear interpolation, which uses a blending function $\alpha\colon \omega_1 \mapsto [0,1]$ to form the join by weighting each component shape appropriately. This join is the region in between the dotted lines on the surface in Figure 3. The formulation for a blended shape is as follows:

$$\mathbf{s}(\mathbf{u}) = \begin{cases} \mathbf{s}_1(\mathbf{u}) & \mathbf{u} \in \Omega_1 \backslash \omega_1 \\ \mathbf{s}_2(\mathbf{u}) & \mathbf{u} \in \Omega_2 \backslash \omega_2 \\ \mathbf{s}_1(\mathbf{u})\alpha(\mathbf{u}) & \\ \quad + \mathbf{s}_2(\beta(\mathbf{u}))(1-\alpha(\mathbf{u})) & \mathbf{u} \in \omega_1 \end{cases} \qquad (3)$$

Defining $\mathbf{s}$ or α for points $\mathbf{u} \in \omega_2$ is possible, but it is unnecessary and redundant, due to the quotienting of Ω^* in (2).

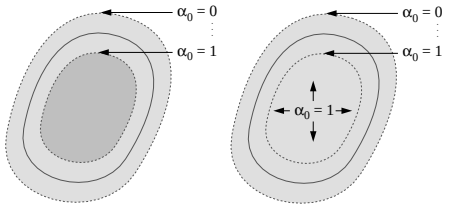
The blending function α has the value 1 where the surface is retained, and 0 where the surface is discarded. For an annulus-shaped region, such as in Figure 6, the blending function α ranges from 0 to 1 from where the surface is discarded to where it is retained, as in Figure 8(a).

For disk-shaped blending regions, α incorporates a parameter determining the degree to which the second component shape is expressed. (This parameter allows smooth evolution of shape). α is therefore defined in terms of a base blending function α_0 and a parameter $h \in [0,1]$:

$$\alpha(\mathbf{u}) = \alpha_0(\mathbf{u})(1-h) + h \qquad \mathbf{u} \in \omega_1 \qquad (4)$$

The parameter h maps the blending function range from $[0,1]$ to $[h,1]$ so that when $h = 0$, $\alpha(\mathbf{u}) = \alpha_0(\mathbf{u})$, and when

$h = 1$, $\alpha(\mathbf{u}) = 1$. In a disk-shaped blending region, such as in Figure 7, α_0 has the value 1 throughout the interior of the disk, as in Figure 8(b).



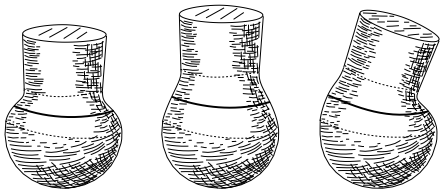
(a) (b)

Figure 8: Blending function α_0 values for different region types

A value of h is specified for each blending region, but for annular blending regions, such as that in Figure 8(a), h is always 0 and $\alpha = \alpha_0$.

3.5 Spatial alignment of components

The relative position and orientation of $\mathbf{s}_1$ and $\mathbf{s}_2$ has a significant effect on the geometry of the join between them. Figure 9 shows the effect on the blended result of rigidly translating and rotating $\mathbf{s}_2$ using the example from Figures 2 and 3, which can be accomplished if $\mathbf{s}_2$ has parameters for rigid translation and rotation. Figure 9(a) shows the original blended shape, the shape in (b) has $\mathbf{s}_2$ translated vertically upward, and the blended shape in (c) has both a translation and a rotation applied.



(a) (b) (c)

Figure 9: Various spatial alignments of $\mathbf{s}_2$

Large discrepancies in the translation or rotation should be avoided, since they are likely to produce bad joins (such as an interpenetration). However, in most cases, the loops generated by $\mathbf{s}_1(\kappa_1)$ and $\mathbf{s}_2(\kappa_2)$ have roughly the same orientation, and are not spatially separated by a large amount. Moreover, it is not difficult to maintain a good join during estimation by putting loose constraints on the rigid motion parameters (which are initially the identity, and produce no discrepancy).

3.6 Hierarchy of blending

Using a blended shape as one of the underlying shapes of another blend produces a hierarchy of blending. If viewed as a directed acyclic graph (dag), its leaves are primitive shapes while the internal nodes of the graph are blended shapes. Note that the dag is not necessarily a tree, based on how holes are added. This dag also provides higher-level topological information about a blended shape, such as part-adjacency information. Such an abstract representation has potential uses in model comparison and object recognition. Section 6 contains experiments demonstrating the stability in the extraction of such a hierarchy.

Using a hierarchy, it is possible to construct any orientable surface. We know from the classification theorem for compact surfaces [42] that the operation of gluing is very powerful, since any orientable surface can be obtained by gluing together flat disks.

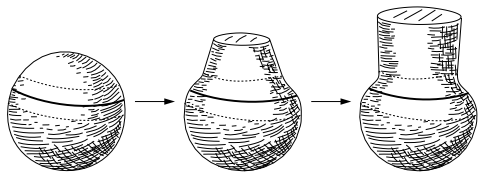
From a shape coverage standpoint, hierarchical blending and CSG produce the same class of shapes. The number of parameters required for a blended shape exceeds that of a CSG model, however, since blending requires the explicit specification of blending region boundaries, while CSG uses object interpenetration. However, it is this explicit specification of where to cut and glue that makes blending useful for shape estimation, which will become clear in the next few sections. The next section describes how such a hierarchy might be formed by repeated application of blending through the process of evolution. After this, Section 5 describes how blending and evolution are used together for shape estimation.

4 Shape evolution

Blending can be used to cut two surfaces apart, and glue selected parts together. Performing these operations on a shape model can be very abrupt—both geometrically and topologically. Shape evolution is concerned with how a shape model changes over time (over the course of fitting). These changes to a shape model include geometric changes (such as a deformation) and representational changes (such as blending the current shape with a second shape). Our goal is to have the changes that occur over time to be geometrically continuous, to achieve greater stability in fitting. Given that topology is a discrete concept, however, it is not possible to produce continuous changes in topology. The next sections describe how parts and holes can be added to the model using evolution.

4.1 Part evolution

As an example, consider the earlier blending example in Figures 2 and 3. Suppose the shape model is initially the sphere (at the top of Figure 2), and the cylindrical part is being added. Instead of abruptly cutting out part of the sphere and gluing-in part of the cylinder, the structure of the shape is *evolved*, and produces a transformation such as that in Figure 10. The blending described in Section 3 can be used to produce such a transformation. There are two operations that can be applied to produce the model transformation in Figure 10 that are useful for smooth model evolution—*splitting* and *gradual blending*. Splitting is a method for adding parts and detail to a model, while gradual blending is used to add a hole or to combine different primitives.



(a) (b) (c)

Figure 10: Part evolution example

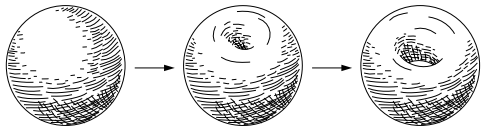
Given an initial model with shape $\mathbf{s}_1$; *splitting* a model involves creating another shape $\mathbf{s}_2 = \mathbf{s}_1$, and replacing the model with a blended shape $\mathbf{s}$ formed from $\mathbf{s}_1$ and $\mathbf{s}_2$. The blending regions of $\mathbf{s}$ replace the removed portion of $\mathbf{s}_1$ with its corresponding region of $\mathbf{s}_2$. Hence, the new model is identical in shape to the initial model, and no geometric discontinuity has occurred during this model transformation. If $\mathbf{s}_1$ is a sphere, the result of a splitting operation could be the shape in Figure 10(a). At this point, the remainder of the transformation in Figure 10(b) and (c) is produced by deforming $\mathbf{s}_2$ into a cylinder.

When the component shapes are not equal, the shapes are *gradually blended* together. Given an initial model with shape $\mathbf{s}_1$ (a sphere), and another shape $\mathbf{s}_2$ (a cylinder); gradually blending-in $\mathbf{s}_2$ involves the specification of a correspondence between the retained portion of $\mathbf{s}_2$, and the discarded portion of $\mathbf{s}_1$. This is specified using a correspondence map β and a disk-shaped blending region such as in Figure 7. Changing the value of the blending parameter h from (4) will alter the contribution that $\mathbf{s}_2$ has in the blended result $\mathbf{s}$. Initially, in Figure 10(a), $h = 1$ so that $\alpha_0 = 1$ and $\mathbf{s} = \mathbf{s}_1$; $\mathbf{s}_2$ has no contribution to the blended result. The contribution of $\mathbf{s}_2$ in the resulting blended shape increases as h decreases from 1 to 0; Figure 10(b) shows the result for $h = \frac{1}{2}$. When $h = 0$, the resulting blended shape has $\mathbf{s}_2$ fully contributed, as in Figure 10(c). It is now possible to replace the disk-shaped blending region with an annular region, as in Figure 8(a). Afterwards, however, the value of h will be restricted to 0.

This seemingly complex process can be automated in a straightforward manner. Section 5.4 describes how the decision to split the model is made. Once the model is split, h (which is one of the shape parameters being estimated) is set to 1. Over the course of fitting, h decreases to 0 (provided the splitting decision was reasonable), and is then held constant. No part of this gradual blending sequence caused a discontinuous jump in the geometry of the shape, which satisfies our goal of smooth evolution.

4.2 Hole evolution

If the evolution involves a topological change to the model, such as the addition of a hole in Figure 11, additional topological surgery must be performed to ensure a smooth transformation. When adding a hole, the key idea is to ensure the shape deforms through an *intermediate shape* which permits a topological change that does not require a geometric change as well. The idea of using intermediate shapes for smooth evolution of shape has also been used for surface metamorphosis [43].



(a) (b) (c)

Figure 11: Hole evolution example

The shape in Figure 11(b) is an intermediate shape between a sphere and torus, sometimes called a "pinched sphere" [44]. From a geometric standpoint, one cannot determine if the shape is a torus with a closed hole or a deformed sphere with two points pushed inward until they met. Figure 12 shows how a pinched sphere is constructed by blending a sphere with two disks removed, with the shape on the right. Topologically, the shape on the right could be two disks or a tube.

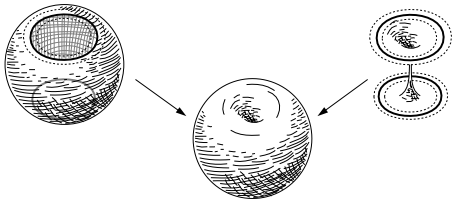


Figure 12: Pinched sphere construction

Adding a hole requires cutting out two pieces of $\mathbf{s}_1$, as in Figure 4. Gluing-in two disks produces a sphere, while

gluing-in a hole, as in Figure 4, produces a torus. By choosing $\mathbf{s}_2$ carefully, a geometrically smooth transition from disks to a hole can be produced.

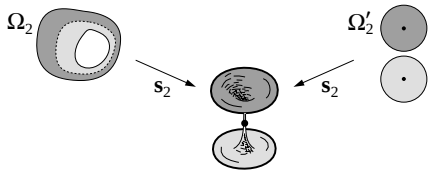


Figure 13: Domains for a "closed" hole

The topological surgery required for this transformation is accomplished by having two domains used to represent Ω_2–one for producing a hole, and one for producing disks. The connection between the two is the intermediate shape, which is when the same geometry is produced by either domain. Figure 13 shows a domain Ω_2 used to produce a hole, and a domain Ω_2' used to produce two disks. The shape in Figure 13 is the retained portion of $\mathbf{s}_2$ used to produce the intermediate shape in Figure 4. Details on the construction of Ω_2 and Ω_2' are given in Appendix B.2. The domain Ω_2' is produced by cutting Ω_2 along the dotted line (into two pieces), and collapsing each dotted path into the center point of a disk. When mapped by $\mathbf{s}_2$, both the dotted path in Ω_2 and the disk center points in Ω_2' map to the pinch-point of the retained portion of $\mathbf{s}_2$ (shown as a small dot on the shape). The hole-adding transformation also requires that the shape $\mathbf{s}_2$ is parameterized in a way that allows the hole to open, as in Figure 14.

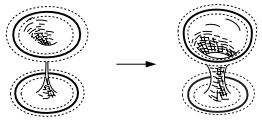


Figure 14: Hole-opening deformation

Using this, a smooth transformation from a sphere to a torus can be produced in three steps. First, two parts of a sphere are removed, such as on the left of Figure 12. This shape is gradually blended with a pinched sphere, shown on the right side of Figure 12, using the two-disk domain Ω_2'. Second, once the pinched sphere is fully expressed, the tube domain of Ω_2 is used in place of Ω_2' (this causes a topological change, but no geometric change). Finally, the torus hole is opened using a deformation such as that in Figure 14.

As with part evolution, this process is also easily automated. Determining the location on the shape, and the time during fitting to apply hole evolution is the subject of Section 5.4. Once the location of the hole blends are determined, the parameter h is used the same way as for part evolution, in order to allow the hole to gradually become expressed. Once h reaches 0 (as before), the hole-size parameters are allowed to vary, which permits the hole to open.

Koenderink [44] provides numerous examples of "morphological scripts" which qualitatively classify shapes based on their formation by evolution. A blending hierarchy (described in Section 3.6) that is formed by evolution provides the same type of information. The process of reconstruction is the subject of the next section, and describes how evolution is realized in this framework.

5 Shape reconstruction

The previous section describes how shape evolution is used in concert with a blended shape representation to smoothly alter the geometry of a shape over time. The driving force for this evolution is a shape reconstruction process.

The shape reconstruction is realized using the physics-based framework of Metaxas and Terzopoulos [1]. This framework is augmented with a decision process that automatically determines if a shape model should be split, as well as if a hole could be added.

Starting from a sphere, the model representation and geometry evolves until a shape that sufficiently represents the data is reached. The lack of any geometric discontinuity over the course of fitting makes the fitting process more robust, since any decision that is made to alter the model representation does not need to consider the effects of a geometric change on the model.

5.1 Deformable model dynamics

Deformable models are parameterized shapes that deform based on a physical model due to forces. For vision applications, deformable models can be used in a physics-based estimation framework [1]. Forces are determined from visual cues such as edges in an image or from geometric information such as range data points. Physics provides additional mathematical tools and is a useful analogy for applications such as shape estimation.

The following gives a brief overview of the dynamic estimation framework from [1]. A shape model $\mathbf{x}$ in this framework is given by:

$$\mathbf{x}(\mathbf{u}) = \mathbf{c} + \mathbf{R}\mathbf{s}(\mathbf{u}) \tag{5}$$

where $\mathbf{c}$ and $\mathbf{R}$ are the global translation and rotation of the model, and $\mathbf{s}(\mathbf{u})$ is a blended shape where $\mathbf{u}$ is taken from the domain Ω. For the applications here, superquadric primitives [45] are used as the component shapes of the blended shape due to their good shape coverage properties

(although other parameterized primitives could have been used).

The deformable shape model is parameterized by a vector of values $\mathbf{q} = (\mathbf{q_c}^\top, \mathbf{q_\theta}^\top, \mathbf{q_s}^\top)^\top$, where $\mathbf{q_c} = \mathbf{c}$ is the translation, and $\mathbf{q_\theta}$ is the quaternion that specifies the rotation matrix $\mathbf{R}$. $\mathbf{q_s}$ contains the parameters for the blended shape $\mathbf{s}$, and is described in Section 5.2.

Estimation of the parameters of the model is based on first order Lagrangian dynamics. As the shape changes, velocities of points on the model are given by:

$$\dot{\mathbf{x}}(\mathbf{u}) = \mathbf{L}(\mathbf{u})\dot{\mathbf{q}} \tag{6}$$

where $\mathbf{L} = \partial\mathbf{x}/\partial\mathbf{q}$ is the model Jacobian [1]. The matrix $\mathbf{L}$ converts the parameter velocities into three-dimensional model velocities.

As is often the case in a deformable model framework in a vision application, a simplified version of the Lagrangian dynamic equations of motion of the model [1] are used. In this case, the resulting equations of motion are:

$$\dot{\mathbf{q}} = \mathbf{f_q} \tag{7}$$

where $\mathbf{f_q}$ are the parameter forces, which are determined from the three-dimensional forces $\mathbf{f}$ that are determined from the data:

$$\mathbf{f_q} = \int_\Omega \mathbf{L}(\mathbf{u})^\top \mathbf{f}(\mathbf{u})\, d\mathbf{u} \tag{8}$$

In this case, the matrix $\mathbf{L}$ converts the three-dimensional data forces $\mathbf{f}$ into forces which directly affect the parameters of the model. For estimation applications using range data, the data forces are determined by applying a force from each data point to the current closest point on the model [1].

The estimation process involves numerically integrating the dynamic equations of motion (7) over time. Upon each iteration, the data forces deform the model. After many iterations, the model comes to rest (the forces either vanish or equilibrate) in a state that closely approximates the data. This solution is the best fitting solution (locally in the parameter space). The quality of the particular solution is affected by the model initialization (described in Section 5.3), and the scheduling order of parameters during the estimation [20, 5].

The solution also can be affected on a more global level, such as in cases where the data is incomplete (when a class of shapes fit the data equally well, but vary where the data is missing). Methods here include biasing the model to have a more symmetric shape [46], as well as adding terms that minimize the volume of the resulting model [20, 8].

5.2 Blended deformable models

Blended shapes are easily incorporated into this framework. The incorporation of evolution involves adding a decision process, and is described in Section 5.4.

The parameters for the blended shape $\mathbf{s}$ are given by:

$$\mathbf{q_s} = (\mathbf{q_{s_1}}^\top, \mathbf{q_{s_2}}^\top, \mathbf{q_b}^\top)^\top \tag{9}$$

where $\mathbf{q_{s_1}}$ and $\mathbf{q_{s_2}}$ are the parameters of the component shapes $\mathbf{s}_1$ and $\mathbf{s}_2$, and $\mathbf{q}_\mathrm{b}$ are the parameters necessary to specify how these component shapes are blended together (details are given in Appendix A.3).

The computation of the model Jacobian $\mathbf{L}$ from (3) requires the Jacobian for a blended shape (a block matrix):

$$\mathbf{L_s}(\mathbf{u}) = \left[\, \alpha(\mathbf{u})\mathbf{L}_{\mathbf{s}_1}(\mathbf{u}) \;\middle|\; (1-\alpha(\mathbf{u}))\mathbf{L}_{\mathbf{s}_2}(\mathbf{u}) \;\middle|\; \mathbf{L}_\mathrm{b}(\mathbf{u}) \,\right] \tag{10}$$

where $\mathbf{L}_{\mathbf{s}_1}$ and $\mathbf{L}_{\mathbf{s}_2}$ are the Jacobians of the component shapes $\mathbf{s}_1$ and $\mathbf{s}_2$, and $\mathbf{L}_\mathrm{b}$ is the Jacobian of the parameters used to specify the blending operation:

$$\mathbf{L}_\mathrm{b}(\mathbf{u}) = \frac{\partial\alpha(\mathbf{u})}{\partial\mathbf{q}_\mathrm{b}}\big(\mathbf{s}_1(\mathbf{u}) - \mathbf{s}_2(\mathbf{u})\big) \tag{11}$$

The computation of $\partial\alpha(\mathbf{u})/\partial\mathbf{q}_\mathrm{b}$ depends on the implementation of the blending function α, and is given in Appendix A.4.

5.3 Initialization

The model is initialized to the best-fit ellipsoid of the data. This places the model with $\mathbf{q_c}$ at the centroid of the data, and with both $\mathbf{q_\theta}$ and the axis-length parameters determined using a linear regression technique such as the matrix of central moments [8].

The initial blended model is topologically a sphere. Given the restriction of having a singularity-free surface parameterization from Section 3.2, a sphere can be produced by gluing together two disks. This is accomplished using blending, where the component shapes are the "top" and "bottom" halves of an ellipsoid. The model in Figure 10(a) illustrates a model formed using this construction (an actual example is shown in Figure 30(b)).

The model should still be treated as a single primitive, however. The component shapes are unified abstractly, with $\mathbf{q}_{\mathbf{s}_1} = \mathbf{q}_{\mathbf{s}_2}$, and the blending parameters $\mathbf{q}_\mathrm{b}$ used to connect the two halves are treated as constants for the model, since this blend is only used to connect the two halves together. This construction produces a shape that is functionally the same as a single ellipsoid (i.e.–it has the same set of parameters), but has a singularity-free surface parameterization. Details on the construction of the surface parameterizations of each half of the ellipsoid are given in Appendix B.

5.4 Model splitting

Evolution involves discrete changes to the representation of the model. The estimation framework of Metaxas and Terzopoulos [1] does not change the model representation, but rather updates the parameters of the current model until a steady state solution is reached—when all forces equilibrate or vanish. In our new framework, locations where to split the model (for part evolution) are determined through analysis of the force distribution once the model reaches this steady state. In addition, the surface is constantly tested for self-proximity, so that any potential self-intersection can become a site for hole evolution. Over the course of estimation, the model shape deforms smoothly as it evolves and fits to the data; this process produces a hierarchical blended shape.

Part evolution is used to split the model to obtain a better fit, once the model reaches a steady state. The location of the blending region on the current model is all that is needed to specify the model split. After this, the model splits into two identical shapes, one on each side of the blending region boundary. The blending region description does not need to be exact, initially, since the blending region boundary will improve as a result of the estimation. What is needed here, is a method to initialize the blending region location to an approximate location of a "part" of the data that is not well represented by the shape model.

Liao and Medioni [47] describe a method which involves fitting a separate model to the "residual data points" (those not being accurately fit by the current model) and isolated parts of the surface (where there are no data points nearby). This decision technique appears to require complete and fairly dense range data. The separate model is then added to or subtracted from the model using CSG techniques.

Our approach involves analysis of the forces in regions of the shape where the forces have equilibrated (but not vanished). Figure 15(a) shows a model (an ellipsoid) fitting to a set of data points, showing the correspondences between the data points and particular locations on the surface. Figure 15(b) shows the net forces that result from the force assignments in (a). Notice how some of the forces pull outward on the model and some pull inward (marked $+$ and $-$ respectively). As in this particular example, the outward and inward pulling forces tend to cluster together in regions on the surface (marked as a grey or black surface region in Figure 15(b)).

A good initial guess for a blending region is along one of the boundaries separating the outward and inward forces. By splitting the model along such a boundary using the part splitting described in Section 4.1, one blended component is pulled outward and the other inward, resulting in a blended model that can better represent the data. The following is a description of how the splitting boundaries are determined, once the forces have reached a steady state.

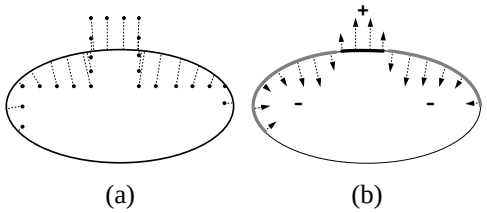


Figure 15: Model forces during equilibration

The boundaries are found using a process analogous to region-growing in images. Instead of images, however, the boundary growing process is performed in the shape domain Ω. The connectivity information (instead of pixel adjacency) is provided by the topology of Ω and the connectivity of the polygon mesh used to represent the shape.

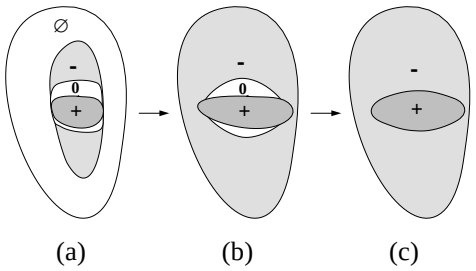


Figure 16: Force regions in the model domain

The shape domain Ω is split into regions based on the direction of the applied forces. At a particular domain point $\mathbf{u} \in \Omega$, the applied force $\mathbf{f}(\mathbf{u})$ is dotted with the corresponding surface normal $\hat{\mathbf{n}}(\mathbf{u})$, as in:

$$F(\mathbf{u}) = \mathbf{f}(\mathbf{u}) \cdot \hat{\mathbf{n}}(\mathbf{u}) \tag{12}$$

The growing is performed using the values $F(\mathbf{u})$ for all $\mathbf{u} \in \Omega$, based on the following classification:

- $F(\mathbf{u}) > \varepsilon_F$ (positive; outward pulling forces)
- $F(\mathbf{u}) < -\varepsilon_F$ (negative; inward pulling forces)
- $|F(\mathbf{u})| \leq \varepsilon_F$ (near zero; equilibrated forces)
- No forces applied (or assigned) to $\mathbf{u}$

where ε_F is a tolerance value related to the amount of noise present in the data. An example domain which has been segmented into regions based on the above categorization is shown in Figure 16(a). The maximum extent of the regions is then computed (if some of the shape is unaffected by forces) by expanding the boundaries of the positive, negative and zero regions, as in Figure 16(b). Finally, the zero regions are excluded by expanding the positive and negative regions, as in Figure 16(c). The resulting regions form a partition of Ω and are each denoted Ω^i.

Pushing the boundaries of the positive and negative regions allows the merging of nearby regions of the same class, as well as the growing of the current regions into unaffected regions of the shape, which tends to maximize the extent of a component part (a reasonable approach given the presence of incomplete data).

Given the example in Figure 16(c), it is reasonable to split the model along the positive/negative boundary, using it as the boundary for a blending region. The problem now has been reduced to deciding along which particular boundary curve to split the shape, since there can be any number of regions, and regions can consist of any number of boundary curves.

Each region is labeled with the amount of total inward or outward force that is applied to the region, as in:

$$f_i = \int_{\Omega^i} F(\mathbf{u}) d\mathbf{u} \tag{13}$$

The quantity f_i for region Ω^i is positive for regions pulled outward, and negative for regions pulled inward. If the applied forces are viewed as a continuous 3D vector field, then f_i is equal to the flux through the surface patch corresponding to Ω^i. Each boundary curve separates two regions Ω^i and Ω^j. The particular boundary curve used for splitting is the one that has the greatest difference—the maximum value of $\left| f_i - f_j \right|$.

The addition of a hole to the model requires gradual blending, and cannot be accomplished using the part splitting mentioned above. The gradual blend of a hole involves the inspection of the model after each fitting iteration, in order to determine if any *non-adjacent* (topologically) locations of the model are within a distance of d_{sep} from each other. In other words, if the surface is about to self-intersect. When these locations are also being pulled toward each other, as in Figure 17(a), the neighborhoods of the closest points are replaced with a hole blend, as in Figure 17(b).

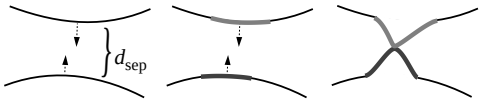


Figure 17: Fitting for hole evolution

Proceeding with fitting at this point may result in the "pinched sphere" shown in Figure 17(c), and is caused by a change in the parameter h. If this is the case, the topology of the shape is changed using the method described in Section 4.2, and the added hole can be opened. Holes which have depth exceeding d_{sep} are added in stages. First, one or two part splitting operations are performed, allowing indentations to be formed. Then, the hole blend is added only after the distance between the indentations drops below d_{sep}. Often, these indentations are later removed from the model since they are no longer being expressed in the final fitted shape (the blending region of the hole increases in size to include the blending region of the indentation part). This method of adding holes requires that there is sufficient data taken from the inside of the hole to "pull" the surface through. The experiments presented in the next section make use of the decision procedures described here, and provide examples of both part splitting and the addition of holes to the model.

6 Experiments and discussion

A number of experiments have been performed using our fully automatic technique for shape estimation from range data. For each experiment, the original set of range data is displayed (sometimes from multiple views to indicate the extent to which the data is incomplete). For some experiments, the range data is taken from the MSU PRIP database [48] or scanned using an available scanner. Other experiments use data generated by a CAD utility which simulates the scanning process on a polygon mesh. Besides showing the model over the course of estimation, the final blending hierarchy is displayed in tree form. Each leaf in the tree corresponds to a particular part that is extracted from the data; the names given to these parts were generated manually, however. During estimation, newly added blending components are first displayed with a white boundary (for clarity).

The first four examples are simple fitting experiments, with the results tabulated in Figure 18. Included in this table are the number of parameters in the final extracted model (the dimension of $\mathbf{q}$), the number of iterations taken, and error measures of the fit. On average, the iterations took just under one second on a 200 MHz SGI Indigo 2, resulting in fitting times on the order of at most a few minutes. The relative RMS error is computed relative to the size of the best fit ellipsoid (the initial fit) of the N data points. The distance is measured between a particular data point $\mathbf{p}_i$, and its corresponding point on the surface $\mathbf{s}(\mathbf{u}_{\mathbf{p}_i})$. This measurement is taken relative to the three vectors d_1, d_2 and d_3 in each principal direction of the best fit ellipsoid (d_1 having the magnitude of the largest diameter, d_3 the smallest). For a particular principal direction j:

$$\text{RMS}_j = \frac{1}{N \left\| d_j \right\|} \sqrt{\sum_{i=1}^{N} \left(d_j \cdot (\mathbf{p}_i - \mathbf{s}(\mathbf{u}_{\mathbf{p}_i})) \right)^2} \quad j = 1,2,3 \tag{14}$$

These experiments are followed by a series of experiments that demonstrate how the extracted models vary

given different range scanner viewpoints, and given small changes in the scanned shape.

Data	Param	Iter	RMS$_j$ ($j = 1,2,3$)
box/cylinder	30	75	0.9%,1.3%,1.8%
mug	69	171	1.2%,1.4%,2.0%
mannequin	180	476	1.5%,2.2%,3.9%
two-holed box	84	291	1.0%,1.4%,2.1%

Figure 18: Experiment data and statistics

The first experiment, shown in Figure 19, shows the fitting of a block and cylinder–a simple part splitting example. The range data, shown from two viewpoints in (a), is from the MSU database. The initial ellipsoid fit is shown in Figure 19(b) with the first model split shown in (c); the greyed region on the model indicates the newly added component. After the split, the fit immediately starts improving, and after a number of iterations, the model in (d) is extracted. The fitting reaches equilibrium at the final model shown in Figure 19(e); the blending hierarchy corresponding to this final model is displayed in (f).

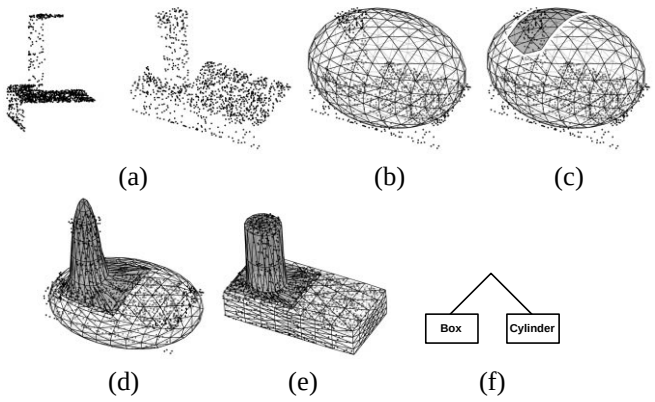


Figure 19: Fitting of a block and cylinder

Figure 20 shows the fitting of range data of a mug, taken from the MSU database. Two views of the data are shown in (a), and the initial ellipsoid fit is shown in (b). The part split in Figure 20(c) leads to a better fit of the mug handle in (d) and (e). The close proximity of either side of the handle in (e) leads to the hole blend added in (f). After a number of iterations, the hole opens, and results in the final extracted mug shape in Figure 20(g), with the blending hierarchy shown in (h). The final shape correctly extends beyond the right of the data (from this viewpoint) in order to match the curvature of the mug. Its extension below the bottom of the data, however, is an artifact of the shape estimation process. Without any means of controlling the volume of the extracted shape, it can grow beyond the bounds of the data. To remedy this, heuristics can be applied which minimize either the volume [20, 8] or description length [30, 31] of the extracted shape.

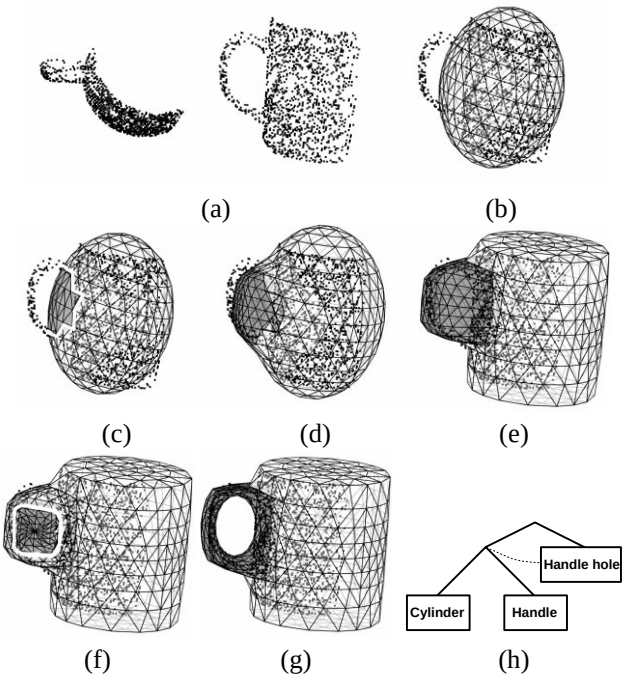


Figure 20: Fitting of a mug

Figure 21 shows the fitting of a fairly complex object; a wooden mannequin. The range data was scanned using an available scanner over a 90 degree wide field-of-view, so that the data is only present on the front of the mannequin. The estimation of the model is shown in (b) through (i); each step shows the newly added blending region, and the fitting of the previous split. The initial ellipsoid fit is shown in Figure 21(b), followed by the results obtained after the first part split in (c) that captures the left arm. Also displayed in (c) is the start of a blend that fits the right arm in (d). This continues showing the fitting of the left leg in (e), the lower and upper parts of the right arm in (f), the right leg and head in (g), the chest in (h), and the lower-torso in (i). The final fit (front and side views) is displayed in (j), with the parts clearly shown in (k). The blending hierarchy in Figure 21(l) shows how each of the added parts are blended onto the body (the original ellipsoid); the structure of the tree also contains information which specifies the order in which the blends took place. It is worth noting that while the fitted shape matches the data very well, the actual shape is somewhat different from the actual shape of the mannequin; especially in the legs. The blending components for the legs are not add-on extremities like the arms, but are raised areas on the torso and body components.

Generated data for a two-holed box (from a single view) is shown in Figure 22(a), with the initial ellipsoid fit shown in (b). The model is split in (c), which allows an indentation to be formed in (d); once this indentation becomes

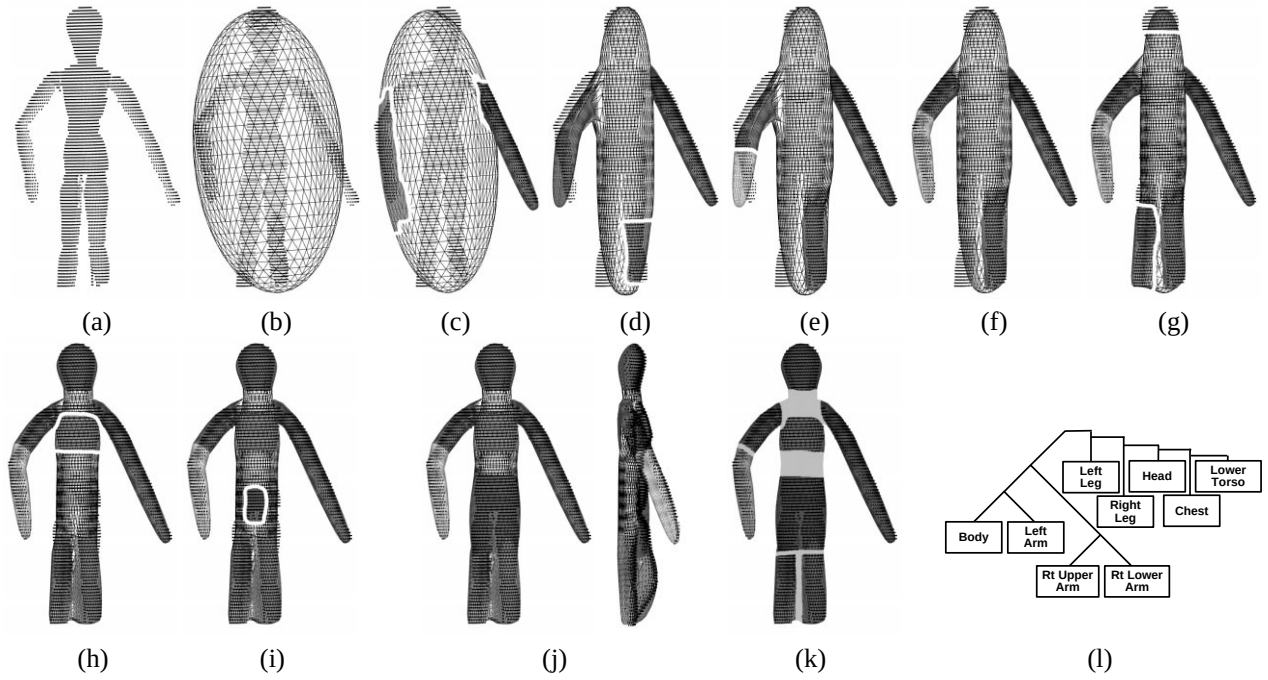


Figure 21: Fitting of a mannequin

deep enough, a hole blend is added in (e), as well as an indentation for the other hole. After more fitting, the hole opens in Figure 22(f), and the other hole indentation deepens. After another hole blend and more fitting, the final model in (g) with the blending hierarchy in (h) is extracted. In the final result, the indentations are no longer expressed in the resulting shape, and were automatically removed.

These four experiments have shown how our framework automatically extracts structured representations of range data using blending. This data can be taken from objects consisting of a number of parts, or taken from objects that have complex topologies.

Figure 23 shows a series of experiments using three sets of data, which were generated using the same object as in Figure 22, but have the scanner in a different position each time. The experiment in Figure 23(a) is the same as in Figure 22. The second experiment in (b) finds a similar shape, except the holes were added to the model in the other order (which produces a different blending hierarchy). In Figure 23(c), the scanner viewpoint did not permit very much data from the hole interiors to be gathered, resulting in a model with a single indentation (and no holes).

From the above set of experiments, it is clear that varying the sensor viewpoint can produce a different extracted structure. Figures 23(a) and (b) produce nearly identical shapes, but the order of the extraction had the holes reversed, with the smaller hole extracted first in (b). This results in the blending hierarchies in (a) and (b) with the big and small holes switched; a simple permutation of the

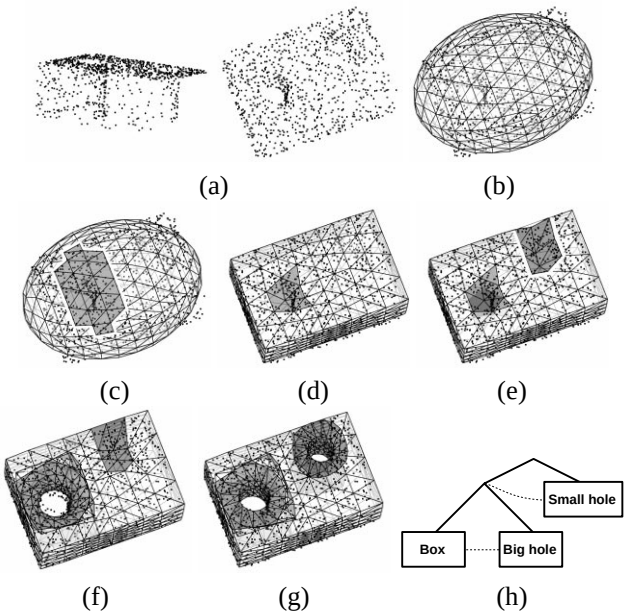


Figure 22: Fitting of a two holed object

resulting hierarchies. Changing the view can also change the structure, as in the extracted shape in Figure 23(c). Because the data did not include information from the hole interiors, only a single indentation formed, instead of any holes. In practice, small changes in viewpoint will often lead to the same qualitative structure being extracted. Of course, there will be some small changes which have large effects. Further investigation of this point is seen in the next set of experiments.

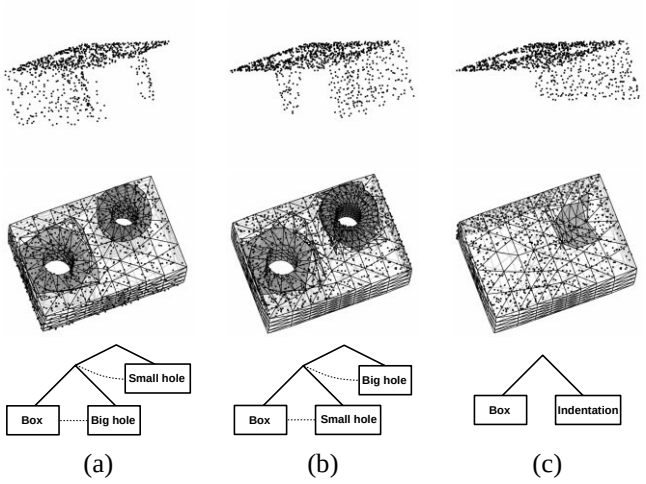


Figure 23: Various 3D range scanner views of two holed object

Figure 24 shows results from a series of 400 fitting experiments using generated data of an object consisting of a box with a hole, and a variable sized cylinder. Fitting results from three of these experiments are shown in Figure 24(a-c), each showing the generated data set, the final fit, and the blending hierarchy. Out of all 400 experiments, only three distinct blending hierarchies resulted. Given the fact that the same parts were extracted across all experiments, and that the "hole" must appear after the "box", these are the only three possibilities, and are exemplified by the blending hierarchies in Figure 24(a-c). The graph in Figure 24(d) indicates the particular blending hierarchy extracted in each experiment, given variation in the height (h_{cyl}) and radius (r_{cyl}) of the cylinder. The letters a, b and c in (d) indicate the particular hierarchy from Figure 24(a-c), with the circled entries being those particular experiments shown in (a-c).

The results from this large set of experiments suggests that our estimation technique is locally stable: a small change in the data is not likely to produce a significantly different extracted shape. This is evident based on the presence of moderately fuzzy boundaries between the solid colored regions in Figure 24(d). The segmentation of the parts of the objects in Figure 24 is relatively simple. Given more complex objects, it is possible that different segmentations will be extracted given the techniques here. Given the same part breakdown, however, a similar extracted structure is guaranteed–the main difference will be permutations in the graph structure of the blending hierarchy.

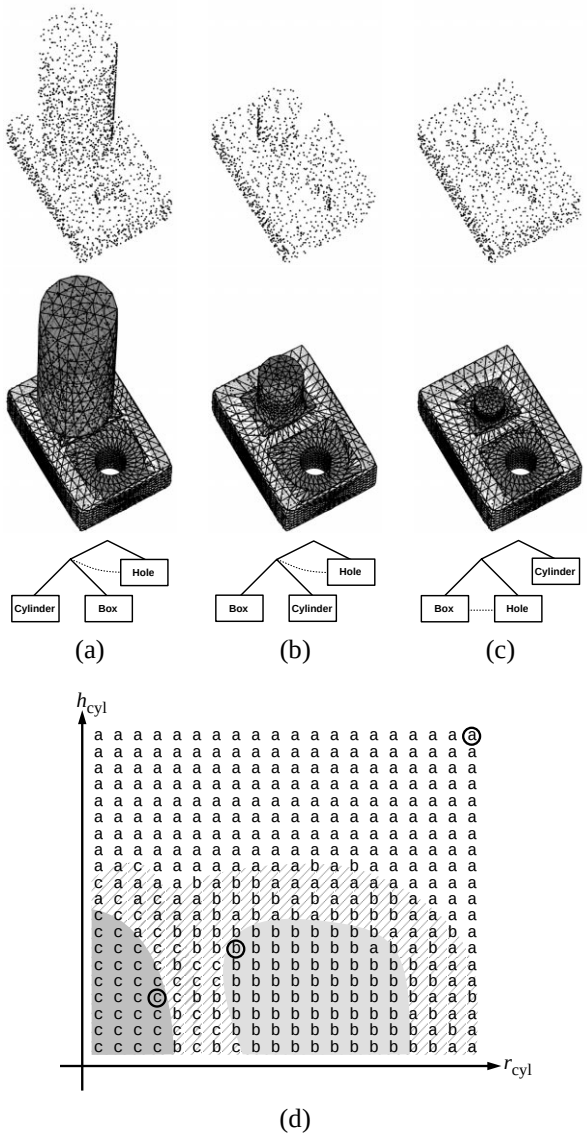


Figure 24: Various cylinder sizes in box/cylinder/hole object

7 Conclusions and future work

We have developed and presented a framework based on shape blending which is used for the shape estimation of incomplete range data for objects of arbitrary topology. The use of this framework has been demonstrated on a variety of examples, which show how a structured representation is extracted in a reasonably stable way. This concise and structured object description allows for use in recognition applications.

Blended shapes along with the process of shape evolution facilitated the realization of a smoothly changing model throughout the process of estimation. A more robust system is obtained by avoiding abrupt changes in the model geometry.

Of course, important work remains. The current decision technique used for the adaptation of the model structure and topology is not based on any notion of the "parts" of the model. It would be worthwhile to incorporate work from part segmentation, which may result in more stable estimation output.

Finally, the topological adaptation of the model could be made more flexible by providing another method for hole evolution. Holes can be added or removed using a "fission" process called torus strangulation [44], where the intermediate shape resembles a croissant. This would allow the framework to back out of a hole decision that seemed reasonable upon its application (in its current form, the framework cannot remove a hole).

Acknowledgments

We would like to thank Eero Simoncelli and Herman Gluck for their helpful discussions. This research is partially supported by NSF Career Award grant 9624604; NSF grant MIP-94-20393; ARO grant DAAH-04-96-1-007; and ONR-YIP grant K-5-55043/3916-1552793.

References

[1] D. Metaxas and D. Terzopoulos, "Shape and nonrigid motion estimation through physics-based synthesis," *PAMI*, vol. 15, no. 6, pp. 580–591, June 1993.

[2] D. Metaxas, *Physics-Based Deformable Models : Applications to Computer Vision, Graphics, and Medical Imaging*, Kluwer Academic Publishers, 1996.

[3] T. Binford, "Visual perception by computer," in *IEEE Conference on Systems and Control*, Dec. 1971.

[4] D. Marr and K. Nishihara, "Representation and recognition of the spatial organization of three-dimensional shapes," *Proceedings Royal Society London*, vol. 200, pp. 269–294, 1978.

[5] T. O'Donnell, T. Boult, X. Fang, and A. Gupta, "The extruded generalized cylinder: A deformable model for object recovery," in *CVPR '94*, 1994, pp. 174–181.

[6] I. Biederman, "Recognition-by-components: a theory of human image understanding," *Psychological Review*, vol. 94, pp. 115–147, Apr. 1987.

[7] A. Pentland, "Perceptual organization and the representation of natural form," *AI*, vol. 28, pp. 293–331, 1986.

[8] F. Solina and R. Bajcsy, "Recovery of parametric models from range images: The case for superquadrics with global deformations," *PAMI*, vol. 12, no. 2, pp. 131–147, 1990.

[9] A. J. Hanson, "Hyperquadrics: smoothly deformable shapes with convex polyhedral bounds," *CVGIP*, vol. 44, pp. 191–210, 1988.

[10] S. Han, D. Goldgof, and K. Bowyer, "Using hyperquadrics for shape recovery from range data," in *ICCV '93*, June 1993, pp. 492–496.

[11] G. Taubin, "An improved algorithm for algebraic curve and surface fitting," in *ICCV '93*, 1993, pp. 658–665.

[12] H. Delingette, "Simplex meshes: A general representation for 3D shape reconstruction," in *CVPR '94*, 1994, pp. 856–859.

[13] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *SIGGRAPH '92*, July 1992, vol. 26, pp. 71–78.

[14] C. Liao and G. Medioni, "Surface approximation of complex multipart objects," in *IEEE Workshop on Physics-Based Modeling in Computer Vision*, 1995, pp. 2–8.

[15] T. McInerney and D. Terzopoulos, "Topology adaptive snakes," *Medical Image Analysis*, to appear.

[16] D. Terzopoulos, A. Witkin, and M. Kass, "Constraints on deformable models: Recovering 3D shape and nonrigid motion," *AI*, vol. 36, no. 1, pp. 91–123, 1988.

[17] R. Malladi, J. Sethian, and B. Vemuri, "Shape modeling with front propagation: A level set approach," *PAMI*, vol. 17, no. 2, pp. 158–175, February 1995.

[18] R. Whitaker, "Algorithms for implicit deformable models," in *ICCV '95*, 1995, pp. 822–827.

[19] R. Szeliski, D. Tonnesen, and D. Terzopoulos, "Modeling surfaces of arbitrary topology with dynamic particles," in *CVPR '93*, 1993, pp. 82–87.

[20] D. DeCarlo and D. Metaxas, "Blended deformable models," in *CVPR '94*, 1994, pp. 566–572.

[21] D. DeCarlo and D. Metaxas, "Blended deformable models," *PAMI*, vol. 18, no. 4, pp. 443–448, April 1996.

[22] D. DeCarlo and D. Metaxas, "Adaptive shape evolution using blending," in *ICCV '95*, 1995, pp. 834–839.

[23] D. Terzopoulos and D. Metaxas, "Dynamic 3D models with local and global deformations: Deformable superquadrics," *PAMI*, vol. 13, no. 7, pp. 703–714, 1991.

[24] A. Pentland and S. Sclaroff, "Closed-form solutions for physically based shape modeling and recognition," *PAMI*, vol. 13, no. 7, pp. 715–729, 1991.

[25] B. C. Vemuri and A. Radisavljevic, "From global to local, a continuum of shape models with fractal priors," in *CVPR '93*, 1993, pp. 307–313.

[26] D.D. Hoffman and W.A. Richards, "Parts of recognition," *Cognition*, vol. 18, pp. 65–96, 1985.

[27] D. Marr, *Vision*, W.H. Freeman, 1982.

[28] A. Gupta and R. Bajcsy, "Volumetric segmentation of range images of 3D objects using superquadric models," *CVGIP*, vol. 58, pp. 302–326, 1993.

[29] P. Besl and R.C. Jain, "Segmentation through variable-order surface fitting," *PAMI*, vol. 10, no. 2, pp. 167–192, March 1988.

[30] T. Horikoshi and S. Suzuki, "3D parts decomposition from sparse range data information criterion," in *CVPR '93*, 1993, pp. 168–173.

[31] A. Leonardis, A. Jaklic, and F. Solina, "Superquadrics for segmenting and modeling range data," *PAMI*, vol. 19, no. 11, pp. 1289–1295, November 1997.

[32] Shigeru Muraki, "Volumetric shape description of range data using "blobby model"," in *SIGGRAPH '91*, July 1991, vol. 25, pp. 227–235.

[33] Y. Sato, J. Ohya, and K. Ishii, "Recovery of hierarchical part structure of 3D shape from range image," in *CVPR '92*, 1992, pp. 699–702.

[34] F. Ferrie, J. Lagarde, and P. Whaite, "Darboux frames, snakes and superquadrics: Geometry from the bottom up," *PAMI*, vol. 15, no. 8, pp. 771–784, 1993.

[35] A. Lejeune and F. Ferrie, "Partitioning range images using curvature and scale," in *CVPR '93*, 1993, pp. 800–801.

[36] K. Siddiqi and B. Kimia, "Parts of visual form: Computational aspects," *PAMI*, vol. 17, no. 3, pp. 239–251, March 1995.

[37] S. Kumar and D. Goldgof, "Model based part segmentation of range data: Hyperquadrics and dividing planes," in *IEEE Workshop on Physics-Based Modeling in Computer Vision*, 1995, pp. 17–23.

[38] V. Guillemin and A. Pollack, *Differential Topology*, Prentice-Hall, 1974.

[39] C. Grimm and J. Hughes, "Modeling surfaces of arbitrary topology using manifolds," in *SIGGRAPH '95*, 1995, pp. 359–368.

[40] L. E. J. Brouwer, *Collected Works Volume 2: Geometry, Analysis, Topology, and Mechanics*, North-Holland Pub. Co., 1976.

[41] M. Rutishauser, M. Stricker, and M. Trobina, "Merging range images of arbitrarily shaped objects," in *CVPR '94*, 1994, pp. 573–580.

[42] W. Massey, *Algebraic Topology: An Introduction*, Springer Verlag, second edition, 1987.

[43] D. DeCarlo and J. Gallier, "Topological evolution of surfaces," in *Graphics Interface '96*, 1996, pp. 194–203.

[44] J. J. Koenderink, *Solid Shape*, MIT Press, 1990.

[45] A. Barr, "Superquadrics and angle-preserving transformations," *IEEE CG&A*, vol. 1, no. 1, pp. 11–23, 1981.

[46] D. Terzopoulos, A. Witkin, and M. Kass, "Symmetry-seeking models and 3D object reconstruction," *IJCV*, vol. 1, no. 3, pp. 211–221, 1987.

[47] C. Liao and G. Medioni, "Simulataneous segmentation and approximation of complex patterns," in *CVPR '94*, 1994, pp. 617–623.

[48] G. Lee and G. Stockman, "Obtaining registered range and intensity images using the Technical Arts scanner," Tech. Rep. CPS-91-08, Dept. of Computer Science, Michigan State University, 1991.

[49] G. Farin, *Curves and Surfaces for Computer Aided Geometric Design*, Academic Press, 1993.

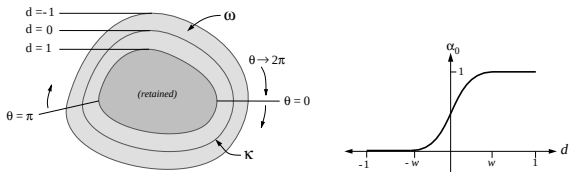
A Blending implementation

This appendix contains details for the implementation of shape blending, such as the representation of blending regions, the construction of the gluing mapping β, and evaluation of the blending function α.

A.1 Blending region parameterization

The mapping β as shown in Figure 6 is constructed by imposing coordinate systems on ω_1 and ω_2, such as C_ω in Figure 25(a) applied to the arbitrary strip ω.

The coordinate system $C_\omega(d,\theta)\colon [-1,1]\times[0,2\pi) \mapsto \omega$ has d measured perpendicular to the curve κ, and θ measured parallel to κ. With the surface cut along κ, and the retained portion shown as a dark gray area in Figure 25(a), $d=-1$ on the boundary of ω where the surface is discarded, $d=0$ along κ, and $d=1$ on the boundary of ω where the surface is retained.



(a) (b)

Figure 25: (a) Strip coordinates C_ω (b) Blending function α_0

A disk-shaped coordinate system used for gradual blending, such as that in Figure 7, is specified in a similar way. The only difference being that the coordinate system is imposed over a disk-shaped region (which requires choosing a disk center point) instead of over an annulus.

Once the coordinate system C_ω is in place, its coordinates are used to evaluate the blending function α, or to construct the correspondence map β. The implementation details on how this coordinate system is constructed are given in Appendix A.2.

Given a point $\mathbf{u}\in\omega$ with strip coordinates (d,θ) so that $\mathbf{u}=C_\omega(d,\theta)$, d is needed to compute the blending function value (for the applications here, the blending function does not depend on θ). A blending function "steepness" parameter $w\in(0,1]$ controls the extent of the 0 to 1 transition in α_0, as in Figure 25(b). The base blending function is defined as:

$$\alpha_0\big(C_\omega(d,\theta)\big)=\begin{cases}0 & d<-w\\ 1 & d>w\\ \dfrac{(d+w)^2(2w-d)}{4w^3} & \text{otherwise}\end{cases} \tag{15}$$

This particular function produces a smooth C^1 join between blending components given the definition of blending in (3).

Using the coordinate systems C_{ω_1} on ω_1 and C_{ω_2} on ω_2, the mapping β, which specifies the topological information for the gluing, is constructed as:

$$\beta\big(C_{\omega_1}(d,\theta)\big)=C_{\omega_2}(-d,\theta)\qquad \begin{matrix}d\in[-1,1]\\ \theta\in[0,2\pi)\end{matrix} \tag{16}$$

This mapping matches up the discarded boundary of one shape with the retained boundary of the other. The orientation of the boundary of the blending components is specified by the direction that θ increases, shown by the arrows in Figure 25(a). In order to produce orientable surfaces, β should preserve the orientation, which should be chosen by some consistent labeling rule (such as a right-hand rule used to point outside the surface). Not doing this can produce closed non-orientable surfaces–a property an estimated shape is not likely to have! The next section describes a method of how such a coordinate system can be placed on a shape.

A.2 Blending region representation

A scheme for representing the blending regions is required for defining the cutting and gluing operations described in Section 3 (the blending curves κ and strips ω). It is also needed for computing the blending function α and correspondence map β. This is accomplished by establishing a coordinate system on the blending regions which can be used to identify specific points.

Figure 26 shows the curve κ inside of a domain Ω. κ is represented in a piecewise manner, consisting of N_κ sections, numbered using the index set

$$I_\kappa=\mathbb{Z}/N_\kappa\mathbb{Z}=[0,\ldots,N_\kappa-1] \tag{17}$$

which produces a numbering of the pieces that wraps around at both ends $\pmod{N_\kappa}$ to accommodate the loop structure of κ. The sections of κ are:

$$\big\{\kappa_{i(i+1)}\,\big|\,i\in I_\kappa\big\} \tag{18}$$

These segments are specified in terms of some points and scalar values:

$$\big\{k_{i(i+1)}\in\Omega\,\big|\,i\in I_\kappa\big\}\qquad\big\{\mu_i\in[0,1]\,\big|\,i\in I_\kappa\big\} \tag{19}$$

From this information, additional points can be computed:

$$k_i=k_{(i-1)i}(1-\mu_i)+k_{i(i+1)}\mu_i\quad i\in I_\kappa \tag{20}$$

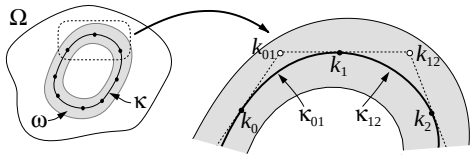
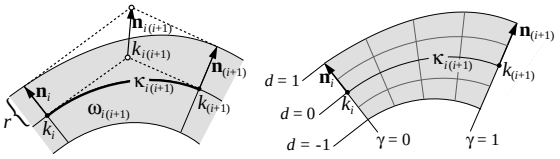


Figure 26: Blending region points

The definition of an arbitrary section is shown in Figure 27(a), which shows how the curve segment $\kappa_{i(i+1)}$ is determined given points k_i, $k_{i(i+1)}$, and $k_{(i+1)}$, as given in (19) and (20). These three points form the control polygon for a quadratic Bézier curve [49]. The curve has endpoints k_i and $k_{(i+1)}$, where the curve is also tangent to the segments of its control polygon, and is given by (in terms of Bernstein polynomials):

$$\begin{aligned}\kappa_{i(i+1)}(\gamma) = \quad & k_i(1-\gamma)^2 \\ & + k_{i(i+1)}2\gamma(1-\gamma) \quad \gamma \in [0,1] \\ & + k_{(i+1)}\gamma^2\end{aligned} \tag{21}$$

The strip $\omega_{i(i+1)}$ that surrounds the curve $\kappa_{i(i+1)}$ is also shown in Figure 27(a), and is determined using normals to the curve $\kappa_{i(i+1)}$. $\mathbf{n}_i$ and $\mathbf{n}_{(i+1)}$ are the normals to the curve segment at its endpoints, each with magnitude r, and point in the direction of the retained portion of the surface. r is a parameter which specifies the width of the blending strip ω, as seen in Figure 27(a). For the applications here, the blending region has a width of r along its entire length.



(a) (b)

Figure 27: Blending region detail and segment coordinates

The strip $\omega_{i(i+1)}$ is defined as the area covered by the Bézier curves defined by the family of control polygons:

$$\left\{\begin{matrix} k_i + d\mathbf{n}_i \\ k_{i(i+1)} + d\mathbf{n}_{i(i+1)} \\ k_{(i+1)} + d\mathbf{n}_{(i+1)} \end{matrix}\right\} \text{ where } d \in [-1,1] \tag{22}$$

Figure 27(a) shows the control polygons where $d = 0$ and $d = 1$ as dotted paths. The vector $\mathbf{n}_{i(i+1)}$ is defined as:

$$\mathbf{n}_{i(i+1)} = \begin{cases} \mathbf{n}_i & \text{if } \mathbf{n}_i = \mathbf{n}_{(i+1)} \\ \dfrac{r\left(\hat{p}_i + \hat{p}_{(i+1)}\right)}{\left\|\hat{\mathbf{n}}_i \times \hat{\mathbf{n}}_{(i+1)}\right\|} & \text{otherwise} \end{cases} \tag{23}$$

where $p_i = k_{i(i+1)} - k_i$ and $p_{(i+1)} = k_{i(i+1)} - k_{(i+1)}$

This definition of $\mathbf{n}_{i(i+1)}$ produces control polygons that are "parallel" to the one used to define $\kappa_{i(i+1)}$. The use of (22) suggests a natural way of imposing a coordinate system on the strip surrounding $\kappa_{i(i+1)}$, seen in Figure 27(b).

The coordinate system $C_{\omega_{i(i+1)}} : [-1,1] \times [0,1] \mapsto \omega_{i(i+1)}$ is imposed using the control polygon given by (22) with γ as the interpolation parameter:

$$\begin{aligned} C_{\omega_{i(i+1)}}(d,\gamma) = \quad & (k_i + d\mathbf{n}_i)(1-\gamma)^2 \\ & + \left(k_{i(i+1)} + d\mathbf{n}_{i(i+1)}\right)2\gamma(1-\gamma) \\ & + \left(k_{(i+1)} + d\mathbf{n}_{(i+1)}\right)\gamma^2 \end{aligned} \tag{24}$$

Determining the (d,γ) coordinates within a blending region segment from a given domain point $\mathbf{u} \in \omega$ requires solving for d and γ in (24). This involves finding the closest point on $\kappa_{i(i+1)}$ to $\mathbf{u}$ to compute γ, after which d can be found using:

$$d = \frac{\left\|\mathbf{u} - \kappa_{i(i+1)}(\gamma)\right\|_{\text{dir}}}{r} \tag{25}$$

where the magnitude is a directed (signed) distance–the positive side of $\kappa_{i(i+1)}$ is in the direction specified by the normals.

The coordinate system $C_\omega(d,\theta)$ is constructed from those for each blending region section. The d value remains unchanged, while the θ value can be computed once its value is defined at each of the k_i with:

$$\{\theta_i \in [0,2\pi) \mid i \in I_\kappa\} \tag{26}$$

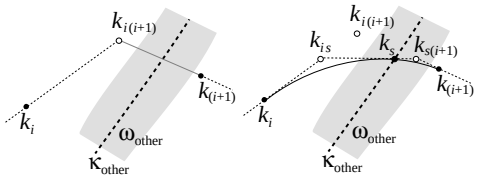
so that the coordinates can be computed as:

$$C_\omega(d,\theta) = C_{\omega_{i(i+1)}}\left(d, \frac{\theta - \theta_i}{\theta_{(i+1)} - \theta_i}\right) \quad \text{for } \theta_i \le \theta < \theta_{(i+1)} \tag{27}$$

once the appropriate segment has been identified to determine i. Because of the convex hull property of Bézier curves [49], the particular segment that contains the parameterization for a domain point $\mathbf{u} \in \omega$ can be found by checking if $\mathbf{u}$ is inside any of the control polygons in (22). This amounts to checking if $\mathbf{u}$ is in the convex hull of the extremal control polygons where $d = \pm 1$.

A.2.1 Blending region crossings

The blending hierarchy described in Section 3.6 leaves the possibility that Ω might be formed by gluing component domains together using (2). The coordinate system described above requires finding paths between different points in Ω. A potential difficulty is illustrated in Figure 28(a), where the path of the blending region crosses the boundary formed by some other blending region ω_{other}.



(a) (b)

Figure 28: Region crossing example

For the case in Figure 28(a), finding the line that connects $k_{i(i+1)}$ and $k_{(i+1)}$ first requires finding the intersection point of this line with κ_{other}. While this could be performed with a simple iterative scheme, it is still a moderately expensive operation. Any computation involving the coordinate system in (24) (which are very frequent) would be very inefficient.

Instead, by subdividing $\kappa_{i(i+1)}$ so that each part lies completely inside of one of the component domains, the coordinate system computations are now efficient. The overlap between the two component domains allows a margin of tolerance of the location of the subdivision point, and also simplifies the construction of the subdivided blending region strip.

This subdivision process is shown in Figure 28(b), and involves performing one iteration of the de Casteljau algorithm [49] to split the control polygon $\{k_i, k_{i(i+1)}, k_{(i+1)}\}$ into $\{k_i, k_{is}, k_s\}$ and $\{k_s, k_{s(i+1)}, k_{(i+1)}\}$ (which does not change the shape of the curve $\kappa_{i(i+1)}$). A suitable subdivision is found by performing a binary search on the interpolating parameter $\gamma \in [0,1]$ using the de Casteljau algorithm to compute k_{is}, $k_{s(i+1)}$ and k_s (by linear interpolation). The search terminates when k_s is inside ω_{other}.

While some of interpolations performed during the computation of k_s are expensive (they involve computing a line that bridges two domains), they are infrequent. The subdivision is precomputed for any domain operations, and is re-evaluated only when the segment changes shape, or when κ_{other} moves a substantial amount (so that k_s moves outside the overlap region ω_{other}).

A.3 Blending parameters

The parameters of a blended shape given in (9) include additional parameters required to specify the blended shape:

$$\mathbf{q}_b = (k_{i(i+1)}, \mu_i, w, h)^\top \mid i \in I_\kappa \tag{28}$$

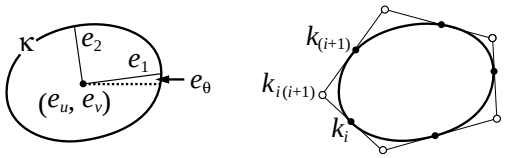
which contains $3N_\kappa + 2$ scalar parameters. This number can be reduced, however, by further parameterization of the shape of the blending regions. The definition of the blending regions given in Appendix A.2 is still used, however. The shape of the blending region is specified by computing $k_{i(i+1)}$ and μ_i in terms of other values.

For example, an ellipse-shaped blending region can be defined as in Figure 29(a) in terms of a center point $(e_u, e_v) \in \Omega$, axis lengths e_1 and e_2, and a rotation e_θ.

Using the θ_i values from (26), the control points k_i and $k_{i(i+1)}$ are computed from the ellipse parameters, as in Figure 29(b). Hence, an ellipse-shaped blending region would have the following parameters:

$$\mathbf{q}_b = (e_u, e_v, e_1, e_2, e_\theta, w, h)^\top \tag{29}$$

which contains 7 scalar parameters. Quadrilateral-shaped blending regions were used in [22].



(a) (b)

Figure 29: Ellipse-shaped blending regions

A.4 Blended shape Jacobian computation

Section 5.1 contains equations which require the computation of Jacobians of blended shapes, such as (10) which requires computing Jacobians $\mathbf{L}_{\mathbf{s}_1}$ and $\mathbf{L}_{\mathbf{s}_2}$ scaled by a blending function term. It also requires the computation of $\mathbf{L}_b$, in (11), which uses the term $\partial\alpha(\mathbf{u})/\partial\mathbf{q}_b$. This can be computed using the chain rule from (4), (15), and (25) as:

$$\frac{\partial\alpha}{\partial\mathbf{q}_b} = \frac{\partial\alpha}{\partial\alpha_0}\frac{\partial\alpha_0}{\partial d}\frac{\partial d}{\partial\mathbf{q}_b} \tag{30}$$

If the blending region shape has been parameterized, as in Figure 29, then $\partial d/\partial\mathbf{q}_b$ will involve an additional chain rule which takes the dependency of the $k_{i(i+1)}$ and μ_i on the region shape parameters.

B Shape primitives

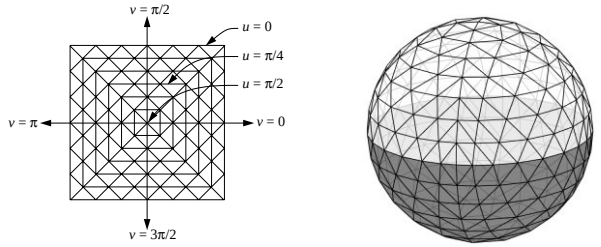
This appendix contains a description of the shape primitives chosen for use here. It contains details of how the ellipsoid surface parameterization was formed, as well as the construction of a hole suitable for shape evolution.

B.1 Ellipsoid parameterization

The ellipsoid (and superquadric ellipsoid [45]) primitives used as the component shapes here use the standard parameterization, as in:

$$\mathbf{s}_{\text{ellipse}}(u,v) = \begin{pmatrix} a_1 \cos u \cos v \\ a_2 \cos u \sin v \\ a_3 \sin u \end{pmatrix} \quad \begin{array}{c} \text{domain } u \in [-\pi/2, \pi/2], v \in [0, 2\pi) \\ \text{parameters } a_1, a_2, a_3 > 0 \end{array} \tag{31}$$

The tessellation of these primitives is somewhat different from what is normally used (which is a latitude-longitude style tessellation). Figure 30(a) shows the tessellation, which is based on a subdivided octahedron. This is to accommodate the singularity free parameterization restriction. The construction of such an ellipsoid (using two halves) as shown in Figure 30(b) was described in Section 5.3.

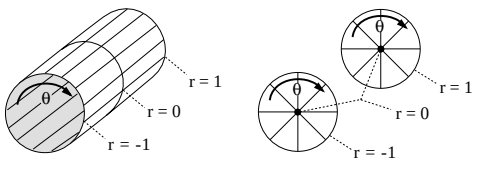


(a) (b)

Figure 30: Split superellipsoid and half of its domain

B.2 Pinched sphere mapping

Section 4.2 discussed the smooth evolution of a hole using gradual blending. The gradual blending of a hole requires the construction of two domains of a pinched sphere, as described in Section 4.2. Figure 31(a) shows the domain used to produce a hole, and (b) shows the domain that produces two disks. At the lowest level, switching between domains amounts to reconnecting mesh nodes [20]. In this case, the nodes along $r = 0$ in the tube in Figure 31(a) are merged together, and then duplicated (and cut apart). This produces the two disks seen in Figure 31(b).



(a) (b)

Figure 31: Pinched sphere domains

The desired shape geometry can be produced by using a parameterized toroid primitive:

$$\mathbf{s}_{\text{torus}}(u,v) = \begin{pmatrix} a_1(a_4 + \cos u)\cos v \\ a_2(a_5 + \cos u)\sin v \\ a_3 \sin u \end{pmatrix} \quad \begin{array}{c} \text{domain } u \in [0, 2\pi), v \in [0, 2\pi) \\ \text{parameters } a_1, a_2, a_3 > 0;\ a_4, a_5 > 1 \end{array} \tag{32}$$

Given the parameterization in Figure 31, the section of the torus in (32) that contains the hole (the blending component of $\mathbf{s}_2$ in Figure 4) is given by:

$$\mathbf{s}_{\text{torus}}\left((1 + \tfrac{r}{2}) \cdot \pi, \theta\right) \quad r \in [-1, 1],\ \theta \in [0, 2\pi) \tag{33}$$

This primitive also produces the hole-opening deformation seen in Figure 14 by increasing a_4 and a_5. A similar parameterization can be performed with a superquadric toroid primitive [45] to allow cylindrical and square holes.